IN THE CLAIMS:

Please amend the claims as follows:

1. (Currently Amended) A method for network packet processing comprises:

receiving network packets at a network processor having multiple engines collectively executing multiple program threads; and

operating on the network packets with a plurality of the program threads to ~~affect processing~~ process the packets.

2. (Original) The method of claim 1 wherein operating comprises:

using at least one program thread to inspect a header portion of the packet.

3. (Original) The method of claim 2 wherein operating further comprises:

signaling by the at least one program thread that a packet header has been processed.

4. (Currently Amended) The method of claim 1, wherein the plurality of program threads comprise ~~are~~ a scheduler program ~~threads~~ thread to schedule task orders for processing and a processing program ~~threads~~ thread that ~~process~~ processes packets in accordance with task assignments assigned by the scheduler program thread ~~threads~~.

5. (Currently Amended) The method of claim 1 wherein ~~each~~ the program threads ~~thread~~ write a message to a register that indicates ~~its~~ the program threads' current ~~status~~ statuses.

6. (Currently Amended) The method of claim 5 wherein interpretation of the message is fixed by a software convention determined between a scheduler program thread and a processing program ~~threads~~ thread called by the scheduler program thread.

7. (Original) The method of claim 5 wherein status messages include busy, not busy, not busy but waiting.

8. (Original) The method of claim 5 wherein a status message includes not busy, but waiting and wherein the status of not busy, but waiting signals that the current program thread has completed processing of a portion of a packet and is expected to be assigned to perform a subsequent task on the packet when data is made available to continue processing of the program thread.

9. (Currently Amended) The method of claim 5 wherein the register is a globally accessible register that can be read from or written to by ~~all current~~ different ones of the program threads executed by different ones of the multiple engines.

10. (Original) The method of claim 4 wherein scheduler program threads can schedule any one of a plurality of processing program threads to handle processing of a task.

11. (Original) The method of claim 10 wherein the scheduler program thread writes a register with an address corresponding to a location of data for the plurality of processing program threads.

12. (Original) The method of claim 11 wherein a selected one of the plurality of processing program threads that can handle the task reads the register to obtain the location of the data.

13. (Original) The method of claim 12 wherein the selected one of the plurality of processing program threads reads the register to obtain the location of the data and to assign itself to processing the task requested by the scheduler program thread.

14. (Original) The method of claim 12 wherein the selected one of the plurality of processing tasks reads the register to obtain the location of the data, while the register is cleared by reading the register by the program thread to assign itself to process the task.

15. (Original) The method of claim 13 wherein when another one of the plurality of processing program threads assignable to the task attempts to read the register after it has been cleared, it is provided with a null value that indicates that there is no task currently assignable to the processing program thread.

16. (Currently Amended) A parallel hardware-based multithreaded processor for receiving network packets comprises:

a general purpose processor that coordinates system functions; and

a plurality of microengines that support multiple program threads, and operate on network packets with a plurality of program threads to ~~affect processing~~ process the packets.

17. (Currently Amended)) The processor of claim 16 wherein one of the plurality of microengines executes scheduler program threads and ~~remaining~~ other ones of the microengines execute processing program threads.

18. (Original) The processor of claim 16 further comprising a global thread status register wherein each program thread writes a message to the global status register that indicates its current status.

19. (Original) The processor of claim 18 wherein interpretation of the message is fixed by a software convention determined between a scheduler program thread and processing program threads called by the scheduler program thread.

20. (Original) The processor of claim 16 further comprising:

a read once register, wherein the scheduler program thread writes the read once register with an address corresponding to a location of data for the plurality of processing program threads and when a selected one of the plurality of processing program threads reads the register to obtain the location of the data, assigns itself to processing the task requested by the scheduler program thread, while the register is cleared by reading the register by the program thread.

21. (Original) The processor of claim 20 wherein when another one of the plurality of processing program threads assignable to the task attempts to read the read once register after it has been cleared, it is provided with a null value that indicates that there is no task currently assignable to the processing program thread.

22. (Currently Amended) An apparatus comprising a machine-readable storage medium having executable instructions for network processing, the instructions enabling the apparatus to:

receive network packets; and

operate on the network packets with a plurality of program threads collectively provided by multiple engines of a network processor to affect processing of the packets.

23. (Original) The apparatus of claim 22 wherein instructions to operate further comprise instructions to:

use at least one program thread to inspect a header portion of the packet.

24. (Original) The apparatus of claim 22 further comprising instructions to provide scheduler program threads to schedule task orders for processing and processing program threads to process packets in accordance with task assignments assigned by the scheduler program threads.

25. (Original) The apparatus of claim 22 wherein each program thread writes a message to a register that indicates its current status.

26. (Original) The apparatus of claim 25 wherein the register is a globally accessible register that can be read from or written to by all current program threads.

27. (Original) The apparatus of claim 22 wherein the scheduler program thread writes a register with an address corresponding to a location of data for the plurality of processing program threads and a selected one of the plurality of processing program threads that can handle the task reads the register to obtain the location of the data, and clears the register after reading by the program thread.

28. (Original) The apparatus of claim 27 wherein when another one of the plurality of processing program threads assignable to the task attempts to read the register after it has been cleared, it is provided with a null value that indicates that there is no task currently assignable to the processing program thread.

29. (New). A system, comprising:

multiple engines of a network processor to execute instructions of multiple threads to process packets received via at least one network interface; and

at least one storage element shared by different ones of the multiple threads to provide inter-thread communication between the threads.

30 (New). The system of claim 29, wherein the at least one storage element comprises a storage element having associated logic to set the register to a reset value after the register is read.

31 (New). The system of claim 30, further comprising:

instructions of a first of the multiple threads, disposed on a computer readable medium, to write a value to the storage element identifying availability of a task; and

instructions of a second of the multiple threads, disposed on a computer readable medium, to read the storage element and perform the task if the retrieved value of the storage element does not equal the reset value of the storage element.

32 (New). The system of claim 29, wherein the at least one storage element comprises a register having different portions allocated to different threads of the different engines.

33 (New). The system of claim 32, further comprising

instructions of a first of the multiple threads, disposed on a computer readable medium, to write a task completion status to the portion allocated to the first of the multiple threads; and

instructions of a second of the multiple threads, disposed on a computer readable medium, to read the task completion status of a portion allocated to the first of the multiple threads.

34 (New). The system of claim 29, further comprising:

instructions of a first of the multiple threads, disposed on a computer readable medium, to set a bit in an array to identify a packet added to a queue.

35 (New). The system of claim 34, further comprising:

instructions of a second of the multiple threads, disposed on a computer readable medium, to determine a queue to service based on the array.

36 (New). The system of claim 29, wherein at least one of the multiple engines comprises an engine having hardware to store execution contexts of multiple ones of the multiple threads.

37 (New). The system of claim 29, further comprising:

instructions of a scheduler thread, disposed on a computer readable medium, to assign a different thread to process a received packet.

38 (New). The system of claim 29, further comprising:

instructions of a receive thread, disposed on a computer readable medium, to reassemble a packet assigned to the thread.

39 (New). The system of claim 29, wherein the storage element comprises a memory internal to the network processor shared by the different engines.

40 (New). A system, comprising:

at least one Ethernet media access controller; and

a network processor communicatively coupled to the at least one Ethernet media access controller to process packets received via the at least one Ethernet media access controller, the processor comprising:

multiple engines to execute instructions of multiple threads, multiple ones of the multiple engines having hardware to store the execution contexts of multiple ones of the multiple threads;

at least one storage element to provide inter-thread communication between the threads; and

thread instructions, disposed on a computer readable medium, to cause at least one of the multiple engines to:

identify a thread to process a received packet;

set a bit within an array to identify a queue for the received packet; and

access the array to identify a queue to service.

41. (New) The system of claim 40, wherein the storage element comprises a memory internal to the network processor, the memory being shared by multiple ones of the multiple engines.

42. (New) The system of claim 40, wherein the storage element comprises at least one of the following: a register having portions assigned to the different threads and a register that resets its value in response to a read access.

43. (New) The method of claim 1, wherein at least one of the engines comprises an engine to execute multiple ones of the multiple threads.

44. (New) The apparatus of claim 22, wherein at least one of the engines comprises an engine to execute multiple ones of the multiple threads.